**Proyecto de Fin de Ciclo Formativo**

**Votaciones Hibernate**

**Iris Núñez Navarro**

**Carlos Galdanez Tarancón**

**¿Qué es Hibernate?**

Para explicar en qué consiste este framework previamente debemos explicar qué es un ORM.

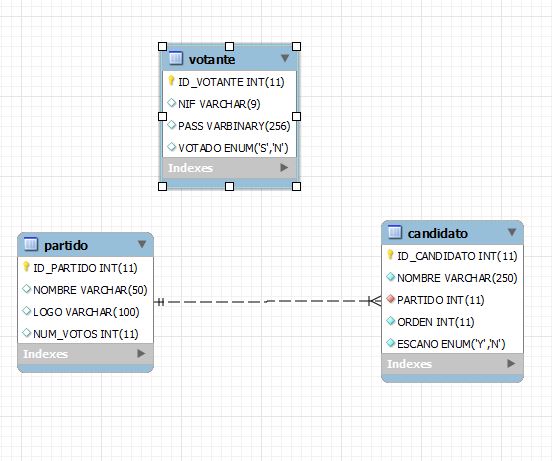
Estas siglas significan “Object-Relational Mapping” (Mapeo objeto-relacional en español) y podríamos decir que un ORM es simplemente un framework que se utiliza para guardar una clase en una base de datos relacional.

Por tanto, Hibernate es u ORM cuya principal particularidad es que no está basado en ningún JSR (Java Specification Requests).

**Implementación del proyecto**

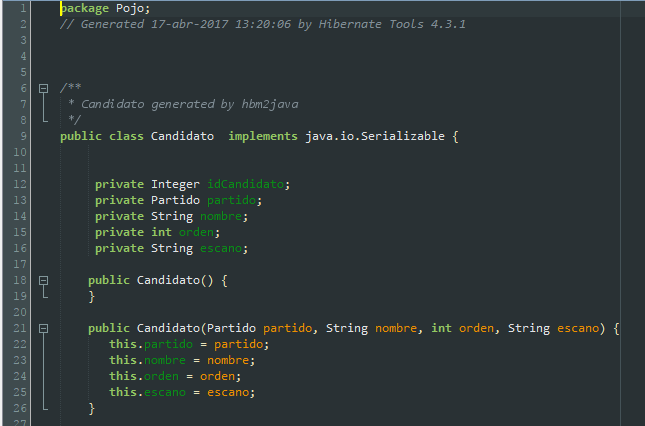
Este proyecto ha sido una modificación del proyecto Votaciones hecho mediante Hibernate durante la segunda evaluación con lo que solo entraremos a especificar la parte del código creada para esta nueva versión.

1º Modificación de la configuración de Hibernate

En la primera versión de este proyecto todas las tablas de la base de datos no estaban relacionadas entre sí, sin embargo, en esta nueva versión se añade la tabla de los candidatos, lo que crea una relación 1:n entre esta tabla y la tabla de los partidos. 

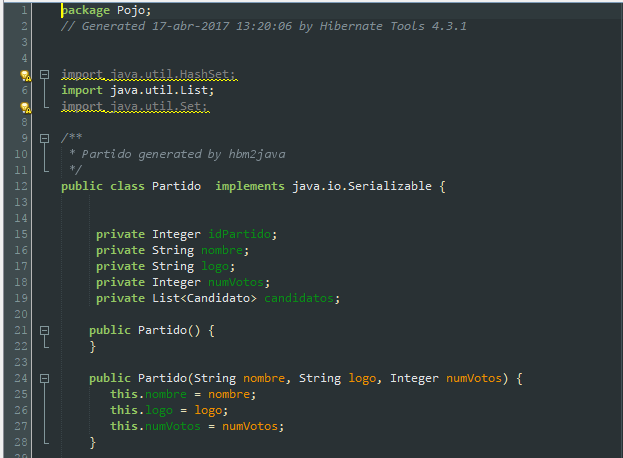
Por lo tanto la estructura de pojos de la versión anterior no nos sirve y debemos crear una nueva. Para esto lo que hemos hecho ha sido generar unos nuevos pojos haciendo una nueva ingeniería inversa y un nuevo mapeo. Tras esto las clases candidato y partido han quedado relacionadas del siguiente modo:

-Clase Candidato

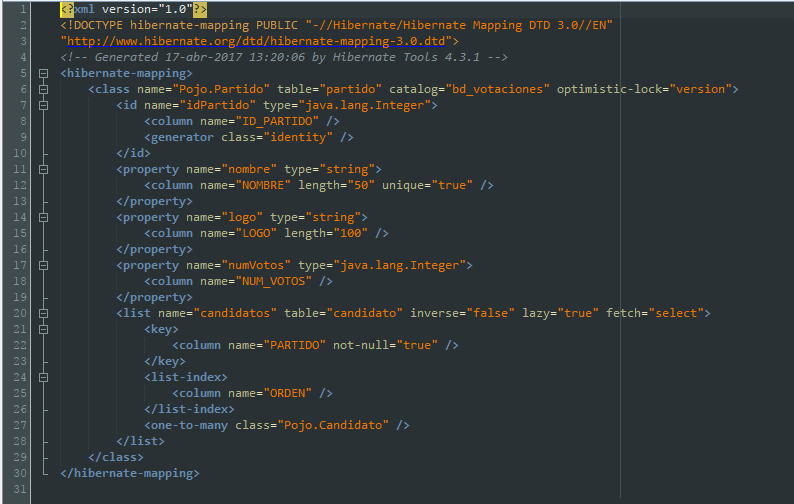


El la clase Candidato se genera un atributo de tipo Partido, en el que se guardará el partido con el que está relacionado el candidato. En el hbm de Candidato podemos observar que este atributo es tipo “many to one”:

-Clase Partido:



En la clase Partido se genera un atributo de tipo List llamado candidatos en el que se almacenarán todos los candidatos relacionados con ese partido.



Inicialmente el atributo candidatos era un tipo Set, pero fue modificado en el hbm para adaptarlo a un tipo List en el que los candidatos saldrán ordenados según el valor “ORDEN”, ya que esto no era posible con el tipo Set.

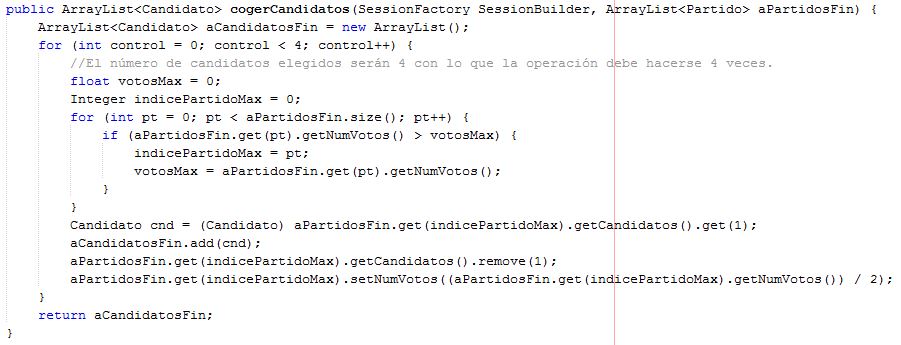
2º Implementación del código

Este es el código del controlador:



De esta nueva versión solo hemos añadido la función cogerCandidatos y la función asignarEscaño cuyo código mostraremos a continuación.

-cogerCandidatos:

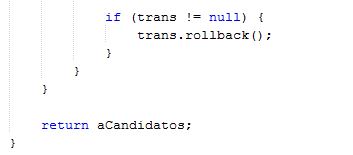


Esta función recibe como parámetro un ArrayList con todos los partidos, de este ArrayList nos interesan dos atributos, el número de votos y sus candidatos.

Como para esta situación el número de escaños será 4 este es el número de veces que se ejecutará el bucle que realiza la siguiente actividad:

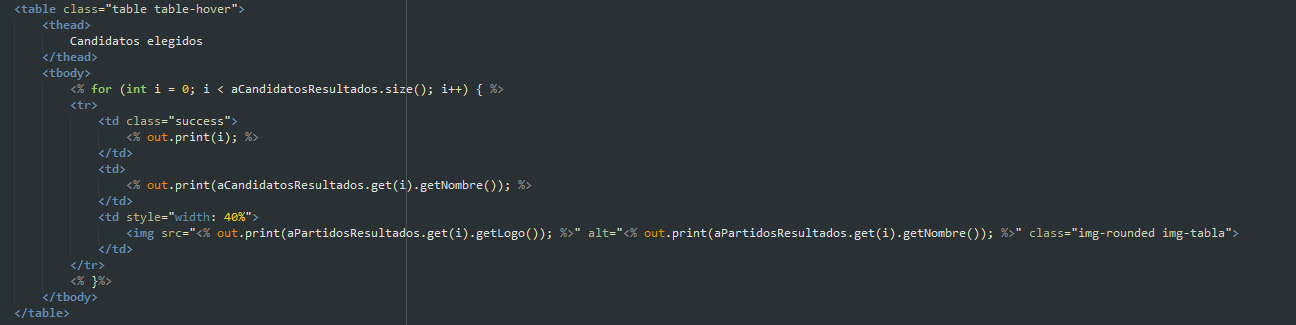
* Primero, recorre todos los partidos que se han presentado y almacena la posición, en el ArrayList que ha recibido como parámetro, del que tenga mayor número de votos así como los votos que tiene.
* Segundo, obtiene el primer candidato del List de candidatos del partido con mayor número de votos (ya se ha explicado previamente el funcionamiento de esta clase), se almacena en la variable que se devolverá al final de la función y se le elimina del List.
* Tercero, se modifica el número de votos del partido ganador dejándolo a la mitad.

-asignarEscaño:



Esta función recibe como parámetro el ArrayList de candidatos que ha generado la anterior función y simplemente cambia el valor del atributo escaño tanto para cada elemento del ArrayList como en la base de datos.

-Vista:



Al fichero .jsp de la versión anterior se le ha añadido el código mostrado en la imagen anterior en el que simplemente se recorre el ArrayList de candidatos anteriormente obtenido, que en este fichero se llama aCandidatosResultado, y se muestra su información.

Para el usuario el resultado final será el siguiente:



